Package 'ngvb'

November 19, 2026

Type Package

Title Latent Non-Gaussian Modelling with 'INLA' and Variational Bayes

Version 0.2.1

Description Diagnoses whether the latent Gaussian assumption of a model fitted with the 'INLA' package is adequate, and fits latent non-Gaussian extensions that relax it. The function ng.check() reports a Bayes-factor sensitivity diagnostic that says whether, and where, the Gaussian assumption is inadequate; ngvb() then fits the non-Gaussian model by a variational-Bayes loop around 'INLA', adding local flexibility only where the data support it. A single unified engine implements the conditional precision Q(theta, V) = D(theta)^T diag(1/V) D(theta) and owns its normalizing constant, and a model registry covers iid, random-walk, autoregressive, Ornstein-Uhlenbeck, intrinsic-CAR (besag), simultaneous-autoregression, 'SPDE'/Matern and seasonal models. Methods follow Cabral, Bolin and Rue (2024) <doi:10.1080/01621459.2023.2296704> and Cabral, Bolin and Rue (2025) <doi:10.1093/jrsssb/qkae107>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports Matrix, methods, stats, graphics, utils, codetools, GIGrvg, ggplot2, patchwork

Suggests INLA, testthat (>= 3.0.0), fmesher, sf, spData, spdep, knitr, rmarkdown, bslib

Additional_repositories https://inla.r-inla-download.org/R/stable

URL https://github.com/rafaelcabral-ai/ngvb, https://rafaelcabral-ai.github.io/ngvb/

BugReports https://github.com/rafaelcabral-ai/ngvb/issues

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website rmarkdown

RoxygenNote 7.3.3

NeedsCompilation no

Author Rafael Cabral [aut, cre, cph]

Maintainer Rafael Cabral <rafael.medeiros.cabral@gmail.com>

Repository CRAN

Date/Publication 2026-11-19 10:50:02 UTC

Contents

areal.plot . . . 2
bayes.factor . . . 3
density.ngvb . . . 4
fitted.ngvb . . . 4
geo.plot . . . 5
jumpts . . . 5
ng.check . . . 6
ngvb . . . 7
ngvb_custom . . . 9
ngvb_operator . . . 11
ngvb_precision . . . 11
ngvb_rgeneric . . . 12
ngvb_sample . . . 12
Orthodont . . . 13
plot.ngvb . . . 14
plot.ngvb.check . . . 14
summary.ngvb . . . 15
summary.ngvb.samples . . . 15

Index **17**

`areal.plot` *Plot areal (regional) data on a map.*

Description

Plot areal (regional) data on a map.

Usage

```
areal.plot(map, data, palette = "YlOrRd", title = "Value")
```

Arguments

map	An ‘sf‘ object with one row per region (its geometry column is used).
data	Numeric vector, one value per region, in the row order of ‘map‘.
palette	A ‘RColorBrewer‘ sequential palette name; the default ‘"YlOrRd"‘ shades low values yellow and high values red.
title	Legend title.

Value

A ‘ggplot‘ object.

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
  areal.plot(nc, nc$BIR74, title = "births")
}
```

---

bayes.factor *Bayes factor of a latent non-Gaussian model against its Gaussian counterpart.*

Description

Returns the marginal-likelihood Bayes factor $Z_{\mathrm{LnGM}}/Z_{\mathrm{LGM}}$ with the mixing variables ‘V‘ integrated out by importance sampling from ‘q(V)‘. A value above one favours the non-Gaussian model; the usual Jeffreys reading is that $\log_{10}$ above 0.5, 1, 2 is substantial, strong, decisive.

Usage

```
bayes.factor(x, ...)

## S3 method for class 'ngvb'
bayes.factor(x, ...)

## S3 method for class 'ngvb.samples'
bayes.factor(x, ...)
```

Arguments

x	An ‘ngvb‘ object (sampled internally) or an ‘ngvb.samples‘ object.
...	For the ‘ngvb‘ method, passed to [ngvb_sample()] (e.g. ‘n.samples‘, ‘seed‘).

Value

A list with ‘BF‘, ‘log10BF‘, ‘logBF‘, and the effective sample size ‘ess‘ of the importance weights (a low ‘ess‘ means more draws are needed).

density.ngvb *Posterior density of a component's non-Gaussianity parameter eta.*

Description

For ‘method = "SVI"‘ this is a kernel density estimate over the GIG variational factor's Monte-Carlo sample; for ‘"SCVI"‘ it is the weighted quadrature grid underlying ‘scvi_update()‘. Either way it is ‘q(eta)‘, the actual (approximate) posterior – not just its mean.

Usage

```
## S3 method for class 'ngvb'
density(x, component = x$comp.names[1L], ...)
```

Arguments

x	An ‘ngvb‘ object.
component	Component name (default: the first).
...	Passed to [stats::density()].

Value

A ‘density‘ object (see [stats::density()]); ‘plot()‘ it directly.

fitted.ngvb *Fitted values of the underlying INLA fit.*

Description

Fitted values of the underlying INLA fit.

Usage

```
## S3 method for class 'ngvb'
fitted(object, ...)
```

Arguments

object	An ‘ngvb‘ object.
...	Ignored.

Value

The ‘summary.fitted.values‘ component of the final INLA fit: a ‘data.frame‘ with one row per element of the linear predictor, holding the posterior summaries of the fitted values. Columns are whichever summaries that fit computed, typically ‘mean‘, ‘sd‘, the requested quantiles (‘0.025quant‘, ‘0.5quant‘, ‘0.975quant‘) and ‘mode‘. ‘NULL‘ if the fit carries no fitted values, which happens when ‘ngvb()‘ was given an INLA fit run without them.

---

geo.plot *Plot geostatistical (point-referenced) data, optionally over a mesh.*

Description

Plot geostatistical (point-referenced) data, optionally over a mesh.

Usage

```
geo.plot(data, mesh = NULL, n = 0, palette = "YlOrRd", title = NULL)
```

Arguments

data	A data frame whose first three columns are longitude, latitude and the value to colour by.
mesh	Optional ‘inla.mesh‘/‘fmesher‘ mesh; its triangulation is drawn underneath.
n	Highlight (enlarge) the ‘n‘ locations with the largest values.
palette	A ‘RColorBrewer‘ sequential palette name (default ‘"YlOrRd"‘).
title	Legend title (defaults to the third column’s name).

Value

A ‘ggplot‘ object.

---

jumpts *Simulated time series with two jumps*

Description

A 100-point series used to illustrate a non-Gaussian RW1 latent process. The underlying signal is smooth except for two sudden jumps, which a Gaussian RW1 over-smooths and a non-Gaussian RW1 accommodates.

Usage

```
jumpts
```

Format

A data frame with 100 rows and 2 columns:

x integer time index, 1..100

y observed value

Examples

```
plot(jumpts)
```

`ng.check` *Check the latent Gaussian assumption of an INLA fit.*

Description

Check the latent Gaussian assumption of an INLA fit.

Usage

```
ng.check(
  fit,
  selection = NULL,
  components = NULL,
  compute.fixed = TRUE,
  compute.random = FALSE,
  plot = TRUE
)
```

Arguments

`fit`	An ‘inla‘ object fitted with ‘control.compute = list(config = TRUE)‘.
`selection`	Optional named list of components to check (default: all random).
`components`	Optional operator overrides (e.g. SPDE), as in [ngvb()].
`compute.fixed`	If ‘TRUE‘, also return the sensitivity of each fixed effect to each component’s non-Gaussianity parameter.
`compute.random`	If ‘TRUE‘, also return, for each checked component, the sensitivity of *its own* posterior mean at every node to its own non-Gaussianity – i.e. how much each node’s fitted value would move under [ngvb()], estimated from the base (Gaussian) fit alone, without actually fitting the non-Gaussian model. This is the same Bayes-factor-sensitivity theory used for ‘sens.fixed‘ (Theorem 2 of Cabral, Bolin & Rue, JRSS-B 2025), applied to the component’s own field instead of a fixed effect: it reuses the cross-covariance already computed for ‘d‘, so it costs one extra matrix multiply per hyperparameter configuration, not a refit. Useful as a preview – e.g. mapped over an SPDE mesh – of where a subsequent ‘ngvb()‘ fit is expected to change the predictions the most, before actually running it.
`plot`	If ‘TRUE‘ (default), draw the diagnostic plots (see [plot.ngvb.check()]): per-index Bayes-factor sensitivity, and the observed overall sensitivity against its Gaussian reference distribution.

Value

An object of class ‘ngvb.check‘. Per component, evaluated at the hyperparameter posterior mode $\hat{\gamma}$: the BF sensitivity ‘s0‘, the per-index contributions ‘d‘, and (Gaussian response) the reference SD ‘sd.ref‘ and ‘p.value‘; the hyperparameter-mixture averages are also kept as ‘s0.mixture‘ / ‘d.mixture‘. Plus ‘sens.fixed‘ if ‘compute.fixed = TRUE‘, and ‘sens.random‘ (a named list, one vector per checked component, aligned with ‘selection‘) if ‘compute.random = TRUE‘.

See Also

[ngvb()]

Examples

```
# Requires INLA, which is not on CRAN; see the package website for the
# rendered output of this and other examples.
## Not run:
set.seed(1); n <- 100
x <- cumsum(rnorm(n, sd = 0.3)); x[50:n] <- x[50:n] + 6
y <- x + rnorm(n, sd = 0.4)
LGM <- INLA::inla(y ~ -1 + f(i, model = "rw1", constr = TRUE),
                  data = data.frame(y = y, i = 1:n),
                  control.compute = list(config = TRUE))
ng.check(LGM)      # small p-value flags departure from latent Gaussianity

## End(Not run)
```

---

ngvb *Fit a latent non-Gaussian model from a fitted INLA (LGM) object.*

Description

Fit a latent non-Gaussian model from a fitted INLA (LGM) object.

Usage

```
ngvb(
  fit,
  selection = NULL,
  components = NULL,
  method = c("SVI", "SCVI"),
  alpha.eta = 2,
  identify.scale = TRUE,
  iter = 30,
  stop.rel.change = 0.001,
  n.sampling = 2000,
  verbose = TRUE
)
```

Arguments

`fit`	An ‘inla‘ object (fitted with ‘control.compute = list(config = TRUE)‘).
`selection`	Optional named list of component indices to extend to non-Gaussianity (default: all random effects).
`components`	Optional named list of operator descriptors overriding auto-detection (required for SPDE: ‘list(s = ngvb_operator("spde", spde = spde))‘). A component listed here is taken as the full prior specification (including its ‘pc.prec‘), so its ‘f()‘-term ‘hyper‘ is not consulted.
`method`	Variational algorithm: ‘"SVI"‘ (structured, mean-field; the default – more reliable) or ‘"SCVI"‘ (structured & collapsed; reaches a fixed point in far fewer iterations, but its collapsed marginal can over-shrink a weak-but-genuine effect all the way to Gaussian, ‘eta ~ 0‘, where SVI holds a small positive value). The two agree when the signal is strong or clearly absent and disagree in the weak-signal regime; prefer the default ‘"SVI"‘ unless you specifically need SCVI’s speed and have checked the two give the same answer on your problem. Regardless of method, when a component’s discrepancies are small but roughly homogeneous across all its indices (no real outlier/inlier split), the fit can drift toward a large, mostly prior-driven eta rather than genuine non-Gaussianity – see ‘alpha.eta‘ and the degeneracy warning this function may emit.
`alpha.eta`	Exponential-PC-prior rate(s) on the non-Gaussianity parameter(s): larger values shrink harder toward the Gaussian model (‘eta = 0‘). If a fit triggers the degeneracy warning, raise this (e.g. by 2-5x) before trusting the result.
`identify.scale`	Enforce the scale-identifiability constraint (default ‘TRUE‘). The conditional precision is invariant under $(\tau, \mathbf{V}) \mapsto (c\tau, c\mathbf{V})$, an exact flat ridge in the likelihood that the VB loop can otherwise walk – letting the mixing variables absorb the overall scale and inflating ‘eta‘ spuriously (most visibly for intrinsic models such as ‘besag‘/‘rw‘). With ‘identify.scale‘ on, each component’s discrepancies are re-anchored so their total matches the Gaussian total each iteration, pinning the scale to ‘tau‘ and leaving ‘eta‘ to respond only to *relative* departures (genuine outliers). Turn off only to reproduce the unconstrained updates.
`iter`, `stop.rel.change`, `n.sampling`, `verbose`	VB controls.

Value

An object of class ‘ngvb‘ with the final INLA ‘fit‘, the mixing vectors ‘V‘, the non-Gaussianity parameters ‘eta‘, and their trajectory.

Priors

The ngvb engine rebuilds each selected component’s structure, so ‘model‘/‘graph‘/‘scale.model‘ on the original ‘f()‘ term are replaced. A precision prior set there (‘hyper = list(prec = ...)‘) *is* honored when it is a ‘pc.prec‘ or ‘loggamma‘ prior — it is translated onto the engine’s precision hyperparameter and a message reports it. Any other family, a fixed precision, or a prior on a secondary hyperparameter (e.g. an AR1 correlation) is dropped with a warning and the operator’s default PC prior is used; to control it, pass the operator via ‘components‘ with an explicit ‘pc.prec = c(U = , alpha =)‘ (the PC prior sets P(sigma > U) = alpha, with sigma = 1/sqrt(precision)).

See Also

[ng.check()], [ngvb_operator()]

Examples

```
# Requires INLA, which is not on CRAN; see the package website for the
# rendered output of this and other examples.
## Not run:
data(jumpts)   # a series with two abrupt jumps
LGM  <- INLA::inla(y ~ -1 + f(x, model = "rw1"), data = jumpts,
                   control.compute = list(config = TRUE))
LnGM <- ngvb(LGM, iter = 10)     # non-Gaussian extension, model auto-detected
summary(LnGM)

## End(Not run)
```

`ngvb_custom` *Define a custom latent non-Gaussian model from a precision "square root".*

Description

Supply the dependency matrix ‘D(theta)‘ (a function of the hyperparameters INLA fits, on the internal/unconstrained scale) and the constant vector ‘h‘. The field then has conditional precision ‘Q(V) = D(theta)^T diag(1/V) D(theta)‘, and ‘V = h‘ recovers the Gaussian model. This turns *any* Gaussian model whose precision factors as ‘Q = D^T diag(1/h) D‘ into a latent non-Gaussian one, without hand-writing an rgeneric.

Usage

```
ngvb_custom(
  D,
  h = 1,
  ntheta,
  theta.initial = rep(0, ntheta),
  logprior = function(theta) sum(stats::dnorm(theta, 0, 3, log = TRUE)),
  rankdef = 0L,
  graph = NULL,
  lognc = NULL
)
```

Arguments

`D`	‘function(theta)‘ returning the sparse dependency matrix ‘D(theta)‘, of dimension ‘m x n‘ (‘m‘ driving-noise terms, ‘n‘ latent nodes).
`h`	constant vector of length ‘m‘ (‘E[V] = h‘; ‘1‘ for discrete models). A scalar is recycled.

ntheta	number of hyperparameters.
theta.initial	initial hyperparameters (internal scale).
logprior	'function(theta)' log prior on the internal scale (including Jacobians). Default: independent 'N(0, 3)'.
rankdef	rank deficiency of 'D^T D' (0 = proper; positive for intrinsic models such as 'm < n' difference operators).
graph	optional sparsity pattern of 'D^T D' (the union over 'theta'); computed from 'D(theta.initial)' if omitted.
lognc	optional analytic normalizing constant 'function(theta, Vinv)' returning '0.5 log|Q| - 0.5 n log(2 pi)'. Supply it when the generic determinant is numerically unstable (e.g. 'D' near-singular, as in SAR/CAR).

Details

Typical workflow: build 'op', fit the Gaussian LGM with 'f(idx, model = ngvb_rgeneric(op))', then call 'ng.check(fit, components = ...)' and 'ngvb(fit, components = ...)'.

Value

An operator descriptor for use in [ngvb()] / [ng.check()] via 'components', or to build a Gaussian fit via [ngvb_rgeneric()].

Note

'D', 'logprior' and 'lognc' are evaluated in a separate R process where only base packages are attached. Data they reference (matrices, vectors) is captured automatically, but any *non-base function* must be namespace-qualified (e.g. 'Matrix::Diagonal', 'stats::dnorm') or written with base R. For instance use 'exp(t)/(1 + exp(t))' rather than 'plogis(t)'.

See Also

[ngvb()], [ng.check()], [ngvb_rgeneric()]

Examples

```
# First differences by hand (an RW1), showing the custom interface:
n  <- 8
D0 <- Matrix::bandSparse(n - 1, n, c(0, 1), list(rep(-1, n - 1), rep(1, n - 1)))
op <- ngvb_custom(D = function(theta) sqrt(exp(theta[1])) * D0,
                  h = 1, ntheta = 1, rankdef = 1)
dim(op$Dfunc(0))
```

ngvb_operator *Construct an ngvb operator descriptor.*

Description

Construct an ngvb operator descriptor.

Usage

```
ngvb_operator(type, ...)
```

Arguments

type	Model type: one of ‘"iid"‘, ‘"rw1"‘, ‘"rw2"‘, ‘"ar1"‘, ‘"sar"‘, ‘"car"‘, ‘"spde"‘, ‘"ou"‘, ‘"seasonal"‘.
...	Model-specific arguments (e.g. ‘n‘ for rw/ar/iid, ‘W‘ for sar/car, ‘loc‘ for ou). For ‘"spde"‘, pass ‘spde = INLA::inla.spde2.pcmatern(mesh, prior.range = , prior.sigma =)‘ (or ‘mesh‘ + ‘prior.range‘ + ‘prior.sigma‘ and ngvb builds it); the non-Gaussian SPDE uses that PC prior on the practical range and marginal SD. Plain ‘inla.spde2.matern()‘ fields are rejected — only the PC-prior parameterization is supported.

Value

An operator descriptor: a list with ‘Dfunc(theta)‘, the constant vector ‘h‘, ‘rankdef‘, and prior/graph metadata.

Examples

```
op <- ngvb_operator("rw1", n = 10)
dim(op$Dfunc(0))       # D(theta): the 9 x 10 first-difference operator
ngvb_precision(op)     # Q = D^T diag(1/h) D  (the RW1 structure matrix)
```

ngvb_precision *Conditional precision matrix Q(theta, V) for an operator (main-process helper).*

Description

Used for validation and for computing the discrepancy statistic d_i.

Usage

```
ngvb_precision(op, theta = op$theta.initial, V = op$h)
```

Arguments

op	Operator descriptor.
theta	Hyperparameters (internal scale). Defaults to ‘op$theta.initial‘.
V	Mixing vector. Defaults to ‘op$h‘ (Gaussian model).

Value

A sparse precision matrix ‘Q(theta, V) = D(theta)^T diag(1/V) D(theta)‘.

---

ngvb_rgeneric *Build an INLA rgeneric model for an ngvb operator at a fixed V.*

Description

Build an INLA rgeneric model for an ngvb operator at a fixed V.

Usage

```
ngvb_rgeneric(op, V = op$h)
```

Arguments

op	An operator descriptor from [ngvb_operator()].
V	Numeric vector of mixing variables (length = nrow of the latent field). The precision uses ‘diag(1/V)‘; pass ‘op$h‘ for the Gaussian model.

Value

An object usable as ‘f(idx, model = <this>)‘ in an INLA formula.

---

ngvb_sample *Sample the mixing variables and refit INLA at each draw.*

Description

Draws ‘n.samples‘ values of the mixing vector ‘V‘ from the variational posterior ‘q(V)‘ of a fitted [ngvb()] object, and refits the conditionally-Gaussian LGM with INLA at each draw. The returned object holds the list of ‘inla‘ fits together with the importance weights needed to treat ‘V‘ as integrated out (see [bayes.factor()], [summary.ngvb.samples()]).

Usage

```
ngvb_sample(object, n.samples = 50, seed = NULL, verbose = interactive())
```

Arguments

object	An ‘ngvb‘ object.
n.samples	Number of posterior draws of ‘V‘ (each is one ‘inla‘ refit).
seed	Optional integer seed for reproducibility.
verbose	Show a progress bar.

Value

An object of class ‘ngvb.samples‘: a list with the ‘inla‘ ‘fits‘, the drawn ‘V‘, per-draw log marginal likelihoods ‘logm‘ and log importance weights ‘logw‘, and the Gaussian-model log marginal likelihood ‘logm.LGM‘.

See Also

[bayes.factor()]

Orthodont *Orthodontic growth data*

Description

Growth measurements on 27 children (16 boys, 11 girls) at ages 8, 10, 12, 14, preprocessed for a linear mixed model with random intercepts and slopes. Used to illustrate a two-component (random intercept + random slope) latent non-Gaussian model.

Usage

```
Orthodont
```

Format

A data frame with 108 rows and 6 columns:

subject subject id (random-intercept index)

subject2 subject id (random-slope index; distinct name required by INLA)

Female indicator for female subjects

time age recentred to 0,1,2,3

tF Female x time interaction

value measured distance

Source

Derived from the `Orthodont` data of Potthoff and Roy (1964), as distributed with the **nlme** package.

Examples

```
summary(Orthodont)
```

plot.ngvb *Diagnostic plots for an ngvb fit.*

Description

For each component two panels are drawn: the mixing weights V_i/h_i (posterior mean with a 90 percent credible interval, and the plug-in value actually refit with INLA), and the posterior density of eta (with its mean and median marked). A final panel shows how eta moved across the variational iterations. The plots are ggplot2 objects assembled with patchwork.

Usage

```
## S3 method for class 'ngvb'
plot(x, ...)
```

Arguments

x	An ‘ngvb‘ object.
...	Ignored.

Value

(Invisibly) the ‘ngvb‘ object; called for the plot it draws.

plot.ngvb.check *Diagnostic plots for a latent-Gaussianity check.*

Description

For each checked component, draws (left) the per-index Bayes-factor sensitivity $d_i(y)$ – spikes locate where the Gaussian assumption is least adequate – and (right, for a Gaussian response) the observed overall sensitivity $s_0 = \sum_i d_i$ against its Gaussian reference distribution; an observed value far in the tail (small p-value) signals latent non-Gaussianity.

Usage

```
## S3 method for class 'ngvb.check'
plot(x, ...)
```

Arguments

x	An ‘ngvb.check‘ object from [ng.check()].
...	Ignored.

Value

A ‘patchwork‘ object combining the per-component diagnostic panels; called mainly for the plot it draws.

`summary.ngvb` *Summary of an ngvb fit: the underlying INLA summaries plus, per component, the non-Gaussianity parameter and the most strongly down-weighted (outlying) indices.*

Description

Summary of an ngvb fit: the underlying INLA summaries plus, per component, the non-Gaussianity parameter and the most strongly down-weighted (outlying) indices.

Usage

```
## S3 method for class 'ngvb'
summary(object, n.flag = 5, ...)
```

Arguments

`object`	An ‘ngvb‘ object.
`n.flag`	Number of top mixing-variable indices to report per component.
`...`	Ignored.

Value

The ‘ngvb‘ object, invisibly; called for the summary it prints.

`summary.ngvb.samples` *Importance-weighted posterior summary of the sampled fits.*

Description

Pools a chosen INLA summary across the sampled fits with the importance weights, giving posterior means and standard deviations for the latent non-Gaussian model (with ‘V‘ integrated out), plus the Bayes factor.

Usage

```
## S3 method for class 'ngvb.samples'
summary(
  object,
  what = c("all", "fixed", "hyperpar", "random"),
  verbose = TRUE,
  ...
)
```

Arguments

`object`	An ‘ngvb.samples‘ object.
`what`	One of ‘"all"‘ (default), ‘"fixed"‘, ‘"hyperpar"‘, or ‘"random"‘ – which INLA summary table(s) to pool. ‘"random"‘ pools ‘summary.random‘ for every component (the latent field itself: one table per component, keyed by node ‘ID‘, e.g. mesh node or time index). ‘"all"‘ shows and returns all three, silently skipping any that don’t apply to this model (e.g. ‘"fixed"‘ on a fixed-effect-free model); random-effect tables are printed as a head (all rows are still returned). Run ‘args(summary.ngvb.samples)‘ or ‘?summary.ngvb.samples‘ to see this list again.
`verbose`	If ‘TRUE‘ (default), print the report as a side effect (as ‘ng.check()‘ does for its plot). Set ‘FALSE‘ to only get the return value back, e.g. when pooling ‘what = "random"‘ into a plot without the table being echoed.
`...`	Ignored.

Value

Invisibly: a single data frame of importance-weighted posterior means/sds for ‘what = "fixed"‘ or ‘"hyperpar"‘; a named list of one such data frame per component (keyed by node ‘ID‘) for ‘what = "random"‘; for ‘"all"‘, a list ‘list(fixed = , hyperpar = , random =)‘ with any inapplicable element ‘NULL‘. Printed as a side effect when ‘verbose = TRUE‘.

Index

∗ **datasets**
    jumpts, 5
    Orthodont, 13

areal.plot, 2

bayes.factor, 3

density.ngvb, 4

fitted.ngvb, 4

geo.plot, 5

jumpts, 5

ng.check, 6
ngvb, 7
ngvb_custom, 9
ngvb_operator, 11
ngvb_precision, 11
ngvb_rgeneric, 12
ngvb_sample, 12

Orthodont, 13

plot.ngvb, 14
plot.ngvb.check, 14

summary.ngvb, 15
summary.ngvb.samples, 15